RafaelReyesCarmona / TinyTrackGPS Public

Code Issues Pull requests Actions Projects Wiki Security

main

TinyTrackGPS / src / Display.cpp

RafaelReyesCarmona Final v0.5 ✓ History

1 contributor

282 lines (251 sloc) | 7.52 KB

```
/*
Display.cpp - A simple track GPS to SD card logger. Display module.
TinyTrackGPS v0.5

Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
All rights reserved.

rafael.reyes.carmona@gmail.com

  This file is part of TinyTrackGPS.

  TinyTrackGPS is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  TinyTrackGPS is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with TinyTrackGPS.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "Display.h"

Display::Display(Display_Type t):_screen(t){
    if (_screen == SDD1306_128X64){
        _width = 16;
        _height = 8;
        _offset = 0;
    } else if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
        _width = 16;
        _height = 2;
        _offset = 0;
    }
}

void Display::start(){
    if (_screen == LCD_16X2){
        #if defined(DISPLAY_TYPE_LCD_16X2)
        // DEFINICION DE CARACTERES PERSONALIZADOS
        byte alt[8] = {
            0b00000100,
            0b00001110,
            0b00011111,
            0b00000100,
            0b00000100,
            0b00000100,
            0b00000100,
            0b00000100,
        };

        byte ant[8] = {
            0b00001110,
            0b00010001,
            0b00010101,
            0b00010001,
            0b00000100,
            0b00000100,
            0b00001110,
            0b00000000,
        };

        byte sd[8] = {
            0b00001110,
            0b00010001,
            0b00011111,
            0b00000000,
            0b00000000,
            0b00010111,
            0b00010101,
            0b00011101,
        };

        byte hourglass_0[8] = {
            0b00011111,
            0b00001110,
            0b00001110,
            0b00000100,
            0b00000100,
            0b00001010,
            0b00001010,
            0b00011111,
        };

        byte hourglass_1[8] = {
            0b00011111,
            0b00001010,
            0b00001110,
            0b00000100,
            0b00000100,
            0b00001010,
            0b00001010,
            0b00011111,
        };

        byte hourglass_2[8] = {
            0b00011111,
            0b00001010,
            0b00001110,
            0b00000100,
            0b00000100,
            0b00001010,
            0b00001110,
            0b00011111,
        };

        byte hourglass_3[8] = {
            0b00011111,
            0b00001010,
            0b00001010,
            0b00000100,
            0b00000100,
            0b00001010,
            0b00001110,
            0b00011111,
        };

        byte hourglass_4[8] = {
            0b00011111,
            0b00001010,
            0b00001010,
            0b00000100,
            0b00000100,
            0b00001110,
            0b00001110,
            0b00011111,
        };
        #endif
        #if defined(DISPLAY_TYPE_LCD_16X2)
        lcd = new LiquidCrystal(RS, ENABLE, D0, D1, D2, D3);
        #endif

        #if defined(DISPLAY_TYPE_LCD_16X2)
        lcd->begin(_width, _height);

        lcd->createChar(0, hourglass_0);
        lcd->createChar(1, hourglass_1);
        lcd->createChar(2, hourglass_2);
        lcd->createChar(3, hourglass_3);
        lcd->createChar(4, hourglass_4);
        lcd->createChar(5, alt);
        lcd->createChar(6, ant);
        lcd->createChar(7, sd);
        #endif
    }

    if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306 = new U8X8_SSD1306_128X64_NONAME_HW_I2C(U8X8_PIN_NONE, SCL, SDA);
        u8x8_SSD1306->begin();
        u8x8_SSD1306->setFont(u8x8_font_7x14B_1x2_r);  //u8g2_font_helvR10_tf
        //u8x8_SSD1306->setFontRefHeightExtendedText();
        //u8x8_SSD1306->setDrawColor(1);
        //u8x8_SSD1306->setFontPosTop();
        #endif
    }
}

void Display::clr(){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        #if defined(DISPLAY_TYPE_LCD_16X2)||(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->clear();
        #endif
    }
    else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306->clear();
        #endif
    }
}

void Display::print(int vertical, int horizontal, const char text[]){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        #if defined(DISPLAY_TYPE_LCD_16X2)||(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->setCursor(vertical, horizontal);
        lcd->print(text);
        #endif
    }
    else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        //u8x8_SSD1306->setCursor(vertical, (horizontal*2));
        //u8x8_SSD1306->print(text);
        u8x8_SSD1306->setCursor(vertical, (horizontal*2));
        this->print(text);
        //u8x8_SSD1306->display();
        #endif
    }
}

void Display::print(int line, const char text[]){
    byte pos = _width -(strlen(text));
    pos = (pos >> 1);
    this->print((int)pos, line, text);
}

void Display::print(const char text[]){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
    #if defined(DISPLAY_TYPE_LCD_16X2)||(DISPLAY_TYPE_LCD_16X2_I2C)
    lcd->print(text);
    #endif
    }
    else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306->print(text);
        u8x8_SSD1306->flush();
        #endif
    }
}

void Display::print(const char text1[], const char text2[]){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        this->print(0, text1);
        this->print(1, text2);
    }
    else if (_screen == SDD1306_128X64) {
        this->print(1, text1);
        this->print(2, text2);
    }
}

void Display::print(const char text1[], const char text2[], const char text3[]){

}

void Display::print(const char text1[], const char text2[], const char text3[], const char text4[]

}

void Display::wait_anin(unsigned int t){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        #if defined(DISPLAY_TYPE_LCD_16X2)||(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->setCursor(15,1);
        lcd->write((byte)t%5);
        #endif
    }
    else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        //char p = 0x2c;
        u8x8_SSD1306->drawString((t%16),6,"-");
        #endif
    }
}
/*
void Display::draw_wait(byte t) {
    #if defined(DISPLAY_TYPE_SDD1306_128X64)
    byte draw_percet;
    if (t == 0) draw_percet = 0b0000;
    else if (t == 1) draw_percet = 0b0001;
    else if (t == 2) draw_percet = 0b1001;
    else if (t == 3) draw_percet = 0b1101;
    else if (t == 4) draw_percet = 0b1111;
    else if (t == 5) draw_percet = 0b1110;
    else if (t == 6) draw_percet = 0b0110;
    else if (t == 7) draw_percet = 0b0010;

    u8x8_SSD1306->drawDisc(_width>>1,_height-8,7,draw_percet);
    u8x8_SSD1306->drawCircle(_width>>1,_height-8,7,U8G2_DRAW_ALL);
    #endif
}
*/
void Display::print_PChar(byte c) {
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
    #if defined(DISPLAY_TYPE_LCD_16X2)||(DISPLAY_TYPE_LCD_16X2_I2C)
    lcd->write(c);
    #endif
    }
}

void Display::splash(int time_delay){
    //#if defined(DISPLAY_TYPE_SDD1306_128X64)
    //u8x8_SSD1306->firstPage();
    //do {
    //#endif
    this->print(NAME, VERSION);
    //#if defined(DISPLAY_TYPE_SDD1306_128X64)
    //} while( u8x8_SSD1306->nextPage() );
    //#endif
    delay(time_delay);
}
```